# 流图连通性

2019年7月30日　　12:27

- 
- Tarjan算法和有向图连通性

一. 强连通分量和Tarjan算法

1. flow graph流图：从一个顶点r出发可达所有其他点的有向图，r为源点
2. 流图中，按从源点出发的dfs搜索树中的父子关系，能分类出以下四种边(x,y)
   i. 树枝边：搜索树中的边，x是y的父结点，dfn[x]<dfn[y]
   ii. 前向边：搜索树外的边，x是y祖先结点，dfn[x]<dfn[y]
   iii. 后向边：搜索树外的边，x是y子孙结点，dfn[x]>dfn[y]
   iv. 横叉边：搜索树外的边，xy无上述关系，dfn[x]>dfn[y]
3. Strongly Connected Component强连通分量SCC：极大强连通子图
   i. 强连通图：任两结点都互相可达的有向图
   ii. 有向环是最简单的强连通图
4. Tarjan算法找有向环：
   i. dfs的同时维护栈，保存当前结点的祖先，或能到达祖先的结点
   ii. 每当发现后向边(x,y)连向x的祖先，则x和祖先间的两条路径成环
   iii. 每当发现横叉边连向一个可到达x祖先结点的y，该三点间的路径成环
5. 追溯值low[x]：x的以下相关结点的最小时间戳
   i. 与x邻接但已在栈中的y
   ii. 从搜索子树上的子结点z出发可达的y
6. 追溯值求法：
   i. 初值low[x]=dfn[x]
   ii. 把x入栈，并标记好x已在栈内
   iii. （扫描每条边(x,y)，若发现邻接点y无dfn，则选y做为搜索子树中x的子结点，选(x,y)为搜索子树上的边，对y做dfs，计算好y搜索子树的low值，再做iv；若y已有dfn，则(x,y)不应是搜索树的边，如果y在栈中，即y是x祖先结点或y可达x祖先结点，则跳转v；否则直接跳转vi）
   iv. 对搜索边(x,y)，low[x]=min(low[x],low[y])（y刚dfs好）
   v. 对非搜索边(x,y)，low[x]=min(low[x],dfn[y])（且y此刻在栈中）
   vi. 若最终low[x]=dfn[x]，说明x子树不可到达其祖先，应清栈直至x出栈
   vii. 总结：用栈记录当前搜索到的强连通分量及祖先。指向子结点的边，用low更新；指向栈内结点的边用dfn更新；不在栈内的已知dfn结点跳过；最后若low[x]仍等于dfn[x]说明x及后入栈的结点自成一个SCC，应出栈
7. 利用追溯值求SCC

```
int n,m,num,tot=1;  //点数，边数，点号，边号
int dfn[MN],low[MN],brdg[MN<<1];
int fst[MN],to[MN<<1],nxt[MN<<1];
inline void add(int x,int y){
     to[++tot] =y, nxt[tot] =fst[x], fst[x] =tot;
```

```
}

int stk[MN],top,ins[MN];  //当前scc及祖先栈，栈顶非空元素指针，在栈标记
int s[MN],scnt;     //s[x] = x所属scc编号。scnt = 找到的scc数量
vector<int>scc[MN];       //scc集合
void tarjan(int x){ //深搜x结点
     low[x] = dfn[x] = ++num;
     ins [ stk[++top] = x ] = 1;
     for(int i= fst[x]; i; i= nxt[i]){
          int y= to[i];
          if(!dfn[y])  //先dfs求好y的low
               tarjan(y),
               low[x]= min(low[x], low[y]);
          else if(ins[y])     //回溯到了能成环的点
               low[x]= min(low[x], dfn[y]);
     }      //遍历完所有后继才判断scc
     if(low[x] == dfn[x]){      //子树不可向上回溯
          ++scnt;             //x的子树自成一scc
          int tp;             //while判断只能使用块外变量
          do{
               tp= stk[top--];     //弹出栈顶元素
               ins[tp] =0;
               s[tp] = scnt;//子树内所有点都在新找到的scc中
               scc[scnt].push_back(tp);
          }while(x!=tp);      //直至x也被弹出栈
     }
}

for__(i,1,n) if(!dfn[i]) tarjan(i);
```

8. SCC缩点：遍历每条边，若所连点在不同SCC中则给这两个SCC编号加边

```
int sfst[MN],sto[MN<<1],snxt[MN<<1],stot=1;
void sadd(int x,int y){
     sto[++stot] =y, snxt[stot] =sfst[x], sfst[x] =stot;
}
for__(x,1,n)
     for(int i= fst[x]; i; i= nxt[i])
          if(s[x]!=s[to[i]])
               sadd(s[x],s[to[i]]);
```

二. 相关问题

1. 有向图必经点和必经边：从S点到T点的所有有向路径都会包含的点x或边(x,y)称为X到Y的必经点或边
   i. 注意环上点也可能必经，所以简单的求SCC或缩点不能处理
   ii. Lenguar-Tarjan算法和Dominator-Tree支配树能O（NlogN）地求指定起点S到任意T的必经点集
   iii. DAG的特殊求法：
      1) 按拓扑序，DP求出S起点到x终点的路径数s[x]
      2) 在反向图求x起点到T终点的路径数t[x]
      3) 若s[x]*t[x]==s[T]（=S到T的总路径数），则x是必经点
      4) 若s[x]*t[y]==s[T]，则(x,y)是必经边
      5) 不过路径数一般是指数级的，很难存得下，直接取模由容易误判，需要

对多个质数取模，来判定是否整数集意义上的相同

iv. 例题：CF GYM 101986F，需要求各边是否是有向图最短路上的必经路

1) 先dij跑起点1的最短路，再对反向边跑终点2为起点的最短路
2) 对每条有向边(x,y)若d[1][x]+w[x][y]+d[y][2]==d[1][2]说明这边是最短路需要的边
3) 易知以上边形成一个DAG，将DAG视作无向图，跑无向图割边，是割边就一定是必经边

2. 2-SAT：N个布尔变量的取值能否同时满足M个约束条件（若Ai=p则Aj=q）

i. 建立2N个节点的有向图，设Ai取0和1分别对应节点号i和i+N，则对每个条件形如“若Ai=p则Aj=q”建立一条有向边(i+p*N,j+q*N)，另外，为了考虑逆否命题再建一条(j+(1-q)*N,i+(1-p)*N)。当i和i+N出现在同一强连通分量，说明出现矛盾，无法满足

ii. 扩展域并查集要求每个关系都是原、逆、否、逆否命题同时出现的，有传递性关系，而有向图可以处理原命题不一定和逆命题同时出现的问题

iii. 注意有向图中没有传递性，可能会出现乍一看矛盾的p间接指向p+N，但这是允许的，只要不在同一强连通分量中即可

iv. 若约束条件形如Ai必须取1，则可以给i到i+N连一条边，表示取0是不可以的；若必须取0，则给i+N到i连一条边

3. 2-SAT问题的赋值

i. 引：每个SCC中一个变量的赋值确定后，其他变量也被确定了

ii. 引：选择零出度的点进行赋值，不会影响其他点

iii. 承：对缩点后的图的反图按入度进行按入度的“自顶向下”拓扑排序即可决定原图的点做按出度的自底向上选择零出度点

iv. 即：对连接属于不同ESCC的点x，y的边(x,y)，在反图中加一条(e[y],e[x])其中e[x]指x所在ESCC

v. 对各val[x]赋初值-1表示未决定，0和1表示取假或真

vi. 其实求ESCC号时的回溯过程的标号即为自底向上按零出度序拓扑排序

vii. #define opp(x) (x+N)%(N<<1)

viii. for__(i,1,N<<1) val[i]= e[i] > e[opp(i)];

- 
- 例题

三. 2SAT模板题（P4782）m行：a取va或b取vb。求能否满足，如何满足

1. a取不到va时，b必取vb；b取不到vb时，a必取va

```
const int MN = 1e6 + 5;
int n,m,num,tot=1;  //点数，边数，点号，边号
int dfn[MN<<1],low[MN<<1];
int fst[MN<<1],to[MN<<2],nxt[MN<<2];
inline void add(int x,int y){
     to[++tot] =y, nxt[tot] =fst[x], fst[x] =tot;
}

int stk[MN<<1],top,ins[MN<<1];   //当前scc及祖先栈，栈顶非空元素指针，在栈标记
```

```
int s[MN<<1],scnt;  //s[x] = x所属scc编号。scnt = 找到的scc数量
void tarjan(int x){ //深搜x结点
     low[x] = dfn[x] = ++num;
     ins [ stk[++top] = x ] = 1;
     for(int i= fst[x]; i; i= nxt[i]){
          int y= to[i];
          if(!dfn[y])  //先dfs求好y的low
               tarjan(y),
               low[x]= min(low[x], low[y]);
          else if(ins[y])     //回溯到了能成环的点
               low[x]= min(low[x], dfn[y]);
     }      //遍历完所有后继才判断scc
     if(low[x] == dfn[x]){      //子树不可向上回溯
          ++scnt;             //x的子树自成一scc
          int tp;             //while判断只能使用块外变量
          do{
               tp= stk[top--];     //弹出栈顶元素
               ins[tp] =0;
               s[tp] = scnt;//子树内所有点都在新找到的scc中
          }while(x!=tp);      //直至x也被弹出栈
     }
}
     scanf("%d%d",&n,&m);
     for_(i,0,m){
          int a,va,b,vb;      //a=va || b=vb
          scanf("%d%d%d%d",&a,&va,&b,&vb);
          add(a+n*(va&1),b+n*(vb^1));
          add(b+n*(vb&1),a+n*(va^1));
     }
     int n2=n<<1;
     for__(i,1,n2) if(!dfn[i]) tarjan(i);
     for__(i,1,n) if(s[i]==s[i+n])    //相反值同色
          return puts("IMPOSSIBLE"), void();
     puts("POSSIBLE");
     for__(i,1,n) printf("%d ",s[i]<s[i+n]);
```

四. CF GYM 101987K：每个人提出三个关于R色B色的假设，问能否都猜对至少两次

1. 猜错任一个时，其他两个必须对；猜对一个时对其他两个无必要联系，不连边

```
add(a[0]+n*(va[0]&1),a[1]+n*(va[1]^1));
add(a[0]+n*(va[0]&1),a[2]+n*(va[2]^1));
add(a[1]+n*(va[1]&1),a[0]+n*(va[0]^1));
add(a[1]+n*(va[1]&1),a[2]+n*(va[2]^1));
add(a[2]+n*(va[2]&1),a[1]+n*(va[1]^1));
add(a[2]+n*(va[2]&1),a[0]+n*(va[0]^1));
```

五. educoder.net/tasks/89qvuox2mhec

1. 题意：有向图缩点后求直径，用拓扑排序，先缩点加边时给y的入度++，给所有缩点图的零入度点入队，再：
2. 
```
while(q.size()){          //“求拓扑排序”顺便求DAG最长路
     int x= q.front();  q.pop();
     for(int i= sfst[x]; i; i= snxt[i]){
          int y= sto[i];
          if(dst[y] < dst[x] +sval[i])
               dst[y]= dst[x] +sval[i],
               ans= max(ans, dst[y]);
          if(--deg[y] ==0)
               q.push(y);}}
```

3. 对比cupygblie9fs是无向图缩点后求直径，两次dfs即可